RATINGS PREDICTION PROJECT

Submitted By:

ASHIKA YASMEEN

ACKNOWLEDGMENT

I would like to express my special gratitude to “Flip Robo” team, who has given me this opportunity to deal with a beautiful dataset and it has helped me to improve my analyzation skills. And I want to express my huge gratitude to Mr. Shubham Yadav(SME Flip Robo), the person who has helped me to get out of all the difficulties I faced while doing the project.

Contents:

INTRODUCTION

1.1 Business Problem Framing:

Rating prediction is a well-known recommendation task aiming to predict a user's rating for those items which were not rated. Predictions are computed from users' explicit feedback, i.e. their ratings provided on some items in the past. Another type of feedback are user reviews provided on items which implicitly express users' opinions on items. Recent studies indicate that opinions inferred from users' reviews on items are strong predictors of user's implicit feedback or even ratings and thus, should be utilized in computation.

The rise in E-commerce has brought a significant rise in the importance of customer reviews. There are hundreds of review sites online and massive amounts of reviews for every product. Customers have changed their way of shopping and according to a recent survey, 70 percent of customers say that they use rating filters to filter out low rated items in their searches. The ability to successfully decide whether a review will be helpful to other customers and thus give the product more exposure is vital to companies that support these reviews. There are two main methods to approach this problem. The first one is based on review text content analysis and uses the principles of natural language process (the NLP method). This method lacks the insights that can be drawn from the relationship between costumers and items. The second one is based on recommender systems, specifically on collaborative filtering, and focuses on the reviewer's point of view.

We have a client who has a website where people write different reviews for technical products. Now they are adding a new feature to their website i.e. the reviewer will have to add stars (rating) as well with the review. The rating is out 5 stars and it only has 5 options available 1 star, 2 stars, 3 stars, 4 stars, 5 stars. Now they want to predict ratings for the reviews which were written in the past and they don't have rating. So we, we have to build an application which can predict the rating by seeing the review.

1.2 Conceptual Background of the Domain Problem

Recommendation systems are an important units in today's e-commerce applications, such as targeted advertising, personalized marketing and information retrieval. In recent years, the importance of contextual information has motivated generation of personalized recommendations according to the available contextual information of users. Compared to the traditional systems which mainly utilize user's rating history, review-based recommendation hopefully provide more relevant results to users. We introduce a review-based recommendation approach that obtains contextual information by mining user reviews. The proposed approach relate to features obtained by analysing textual reviews using methods developed in Natural Language Processing (NLP) and information retrieval discipline to compute a utility function over a given item. An item utility is a measure that shows how much it is preferred according to user's current context. In our system, the context inference is modelled as similarity between the user's reviews history and the item reviews history. As an example application, we used our method to mine contextual data from customer's reviews of technical products and use it to produce review-based rating prediction. The predicted ratings can generate recommendations that are item-based and should appear at the recommended items list in the product page. Our evaluations (surprisingly) suggest that our system can help produce better prediction rating scores in comparison to the standard prediction methods.

As far as we know, all the recent works on recommendation techniques utilizing opinions inferred from user's reviews are either focused on the item recommendation task or use only the opinion information, completely leaving user's ratings out of consideration. The approach proposed in this report is filling this gap, providing a simple, personalized and scalable rating prediction framework utilizing both ratings provided by users and opinions inferred from their reviews. Experimental results provided on dataset containing user ratings and reviews from the real world Amazon and Flipkart Product Review Data show the effectiveness of the proposed framework.

1.3 Review of Literature

In real life, people's decision is often affected by friends action or recommendation. How to utilize social information has been extensively

studied. Yang et al. propose the concept of “Trust Circles” in social network based on probabilistic matrix factorization. Jiang et al. propose another important factor, the individual preference. some websites do not always offer structured information, and all of these methods do not leverage user’s unstructured information, i.e. reviews, explicit social networks information is not always available and it is difficult to provide a good prediction for each user. For this problem the sentiment factor term is used to improve social recommendation.

The rapid development of Web 2.0 and e-commerce has led to a proliferation in the number of online user reviews. Online reviews contain a wealth of sentiment information that is important for many decision-making processes, such as personal consumption decisions, commodity quality monitoring, and social opinion mining. Mining the sentiment and opinions that are contained in online reviews has become an important topic in natural language processing, machine learning, and Web mining.

1.4 Motivation for the Problem Undertaken

The project was first provided to me by FlipRobo as a part of the internship program. The exposure to real world data and the opportunity to deploy my skillset in solving a real time problem has been the primary objective.Many product reviews are not accompanied by a scale rating system, consisting only of a textual evaluation. In this case, it becomes daunting and time-consuming to compare different products in order to eventually make a choice between them. Therefore, models able to predict the user rating from the text review are critically important. Getting an overall sense of a textual review could in turn improve consumer experience. However, the motivation for taking this project was that it is relatively a new field of research. Here we have many options but less concrete solutions. The main motivation is to build a prototype of online hate and abuse review classifier which can used to classify hate and good comments so that it can be controlled and corrected according to the reviewer’s choice.

2.Analytical Problem Framing

2.1 Mathematical/ Analytical Modelling of the Problem

In this perticular problemthe Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer. So clearly it is a multi classification problem and I have to use all classification algorithms while building the model. We would perform one type of supervised learning algorithms: Classification. Here, we will only perform classification. Since there only 1 feature in the dataset, filtering the words is needed to prevent overfit. In order to determine the regularization parameter, throughout the project in classification part, we would first remove email, phone number, web address, spaces and stops words etc. In order to further improve our models, we also performed TFID in order to convert the tokens from the train documents into vectors so that machine can do further processing.I have used all the classificationalgorithms while building model then tuned the best model and saved the best model.

2.2 Data Sources and their formats

The data set contains nearly 1,14,491 samples with 3 features. Since **Ratings** is my target column and it is a categorical column with 5 categories so this problem is a **Multi Classification Problem**.The Ratings can be 1, 2, 3, 4 or 5, which represents the likely ness of the product to the customer.The data set includes:

- Review_Title : Title of the Review.
- Review_Text : Text Content of the Review.
- Ratings : Ratings out of 5 stars.

This project is more about exploration, feature engineering and classification that can be done on this data. Since the data set is huge and includes multi-class classification of ratings, we can do good amount of data exploration and derive some interesting features using the Review column available. We need to build a model that can predict Ratings of the reviewer.

2.3 Data Pre-processing

Data pre-processing is the process of converting raw data into a well readable format to be used by Machine Learning model. Data pre-processing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn; therefore, it is extremely important that we pre-process our data before feeding it into our model.

I have used following pre-processing steps:

- Importing necessary libraries and loading dataset as a data frame.
- Checked some statistical information like shape, number of unique values present, info, null values, value counts etc.
- Checked for null values and I replaced those null values using imputation method. And removed Unnamed: 0.
- Visualized each feature using seaborn and matplotlib libraries by plotting distribution plot and word cloud for each ratings.
- Done text pre-processing techniques like Removing Punctuations and other special characters, Splitting the comments into individual words, Removing Stop Words, Stemming and Lemmatization.
- After getting a cleaned data used TF-IDF vectorizer. It'll help to transform the text data to feature vector which can be used as input in our 6 modelling. It is a common algorithm to transform text into numbers. It measures the originality of a word by comparing the frequency of appearance of a word in a document with the number of documents the words appear in. Mathematically, TF-IDF = TF(t*d)*IDF(t,d)
- Balanced the data using SMOTE method.

2.4 Data Inputs- Logic- Output Relationships

The dataset consists of 2 features with a label. The features are independent and label is dependent as our label varies the values(text) of our independent variables changes.

- I checked the distribution of skewness using dist plots and used count plots to check the counts available in each column as a part of univariate analysis.
- Got to know the frequently occuring and rare occuring word with the help of count plot and was able to see the words in the Review text with reference to thier ratings using word cloud.

2.5 Hardware & Software Requirements & Tools Used

The needed time to train the model depends on the capability of the used system during the experiment. Some libraries use GPU resources over the CPU to take a shorter time to train a model. While taking up the project we should be familiar with the Hardware and software required for the successful completion of the project. Here we need the following hardware and software.

Operating System	Windows10
Processor	CORE i3
RAM	16GB

Language: Python

- Python is widely used in numeric and scientific computing.
- Scipy is a collection of packages for mathematics, science and engineering.
- Pandas is a data analysis and modelling library.

3.Data Analysis and Visualization

3.1 Identification of possible problem-solving approaches

I have converted text into feature vectors using TF-IDF vectorizer and separated our feature and labels. Also, before building the model, I made sure that the input data is cleaned and scaled before it was fed into the machine learning models. Just making the Reviews more appropriate so that we'll get less word to process and get more accuracy. Removed extra spaces, converted email address into email keyword, and phone number etc. Tried to make Reviews small and more appropriate as much as possible.

3.2 Algorithms

In this nlp based project we need to predict Ratings which is a multi-class classification problem. I have converted the text into vectors using TFIDF vectorizer and separated our feature and labels then build the model using One Vs Rest Classifier. Among all the algorithms which I have used for this purpose I have chosen SVC as best suitable algorithm for our final model as it is

performing well compared to other algorithms while evaluating with different metrics I have used following algorithms and evaluated them

- LogisticRegression
- DecisionTreeClassifier
- KNNeighborsClassifier
- SupportVectorClassifier
- MultinomialNB
- RandomForestClassifier
- AdaBoostClassifier
- GradientBoostingClassifier
- XGBClassifier
- SGDClassifier

From all of these above models SupportVectorClassifier was giving me good performance with less difference in accuracy score and cv score.

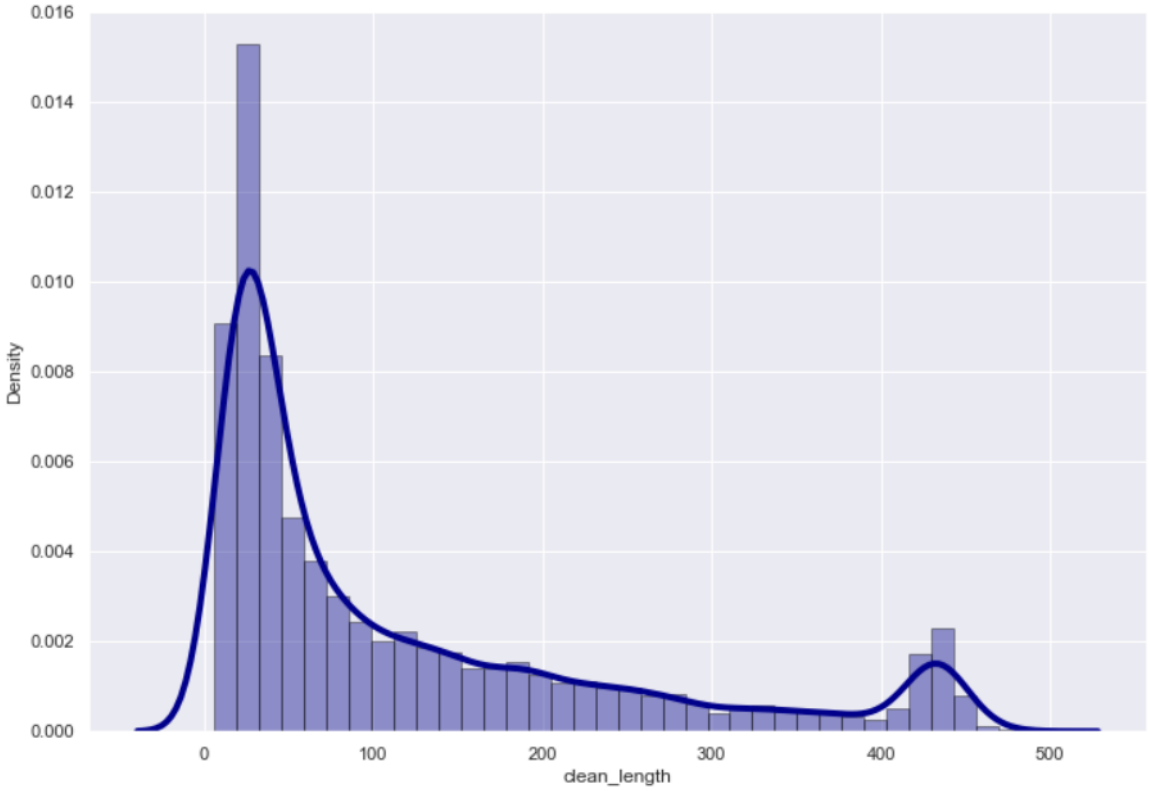
3.3 Key Metrics for success in solving problem under consideration

I have used the following metrics for evaluation:

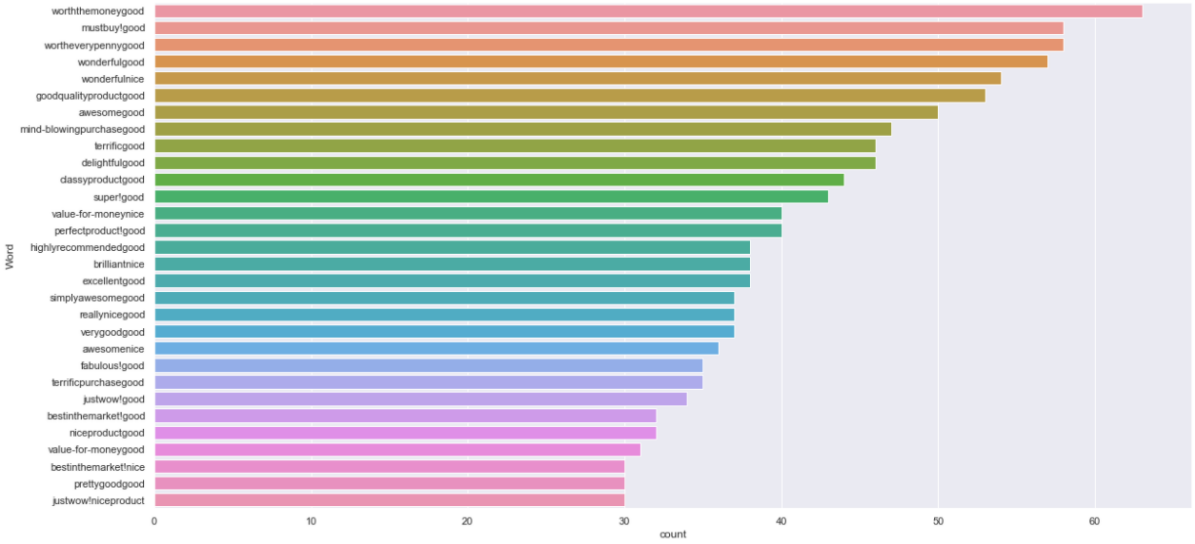
- I have used f1_score, precision_score, recall_score, multilabel_confusion_matrix and hamming loss all these evaluation metrics to select best suitable algorithm for our final model.
- **Precision**can be seen as a measure of quality,higher precision means that an algorithm returns more relevant results than irrelevant ones.
- **Recall** is used as a measure of quantity and high recall means that an algorithm returns most of the relevant results.
- **Accuracy score** is used when the True Positives and True negatives are more important.Accuracy can be used when the class distribution is similar.
- **F1-score** is used when the False Negatives and False Positives are crucial. While F1-score is a better metric when there are imbalanced classes.

3.4 Visualizations

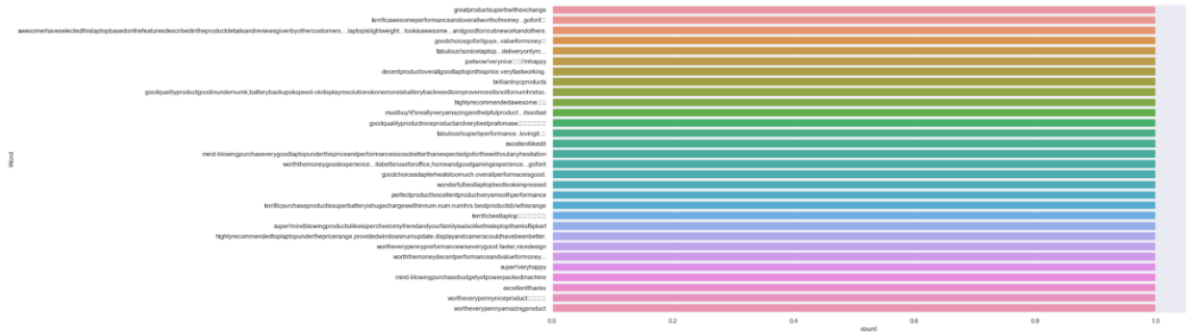
Character Count



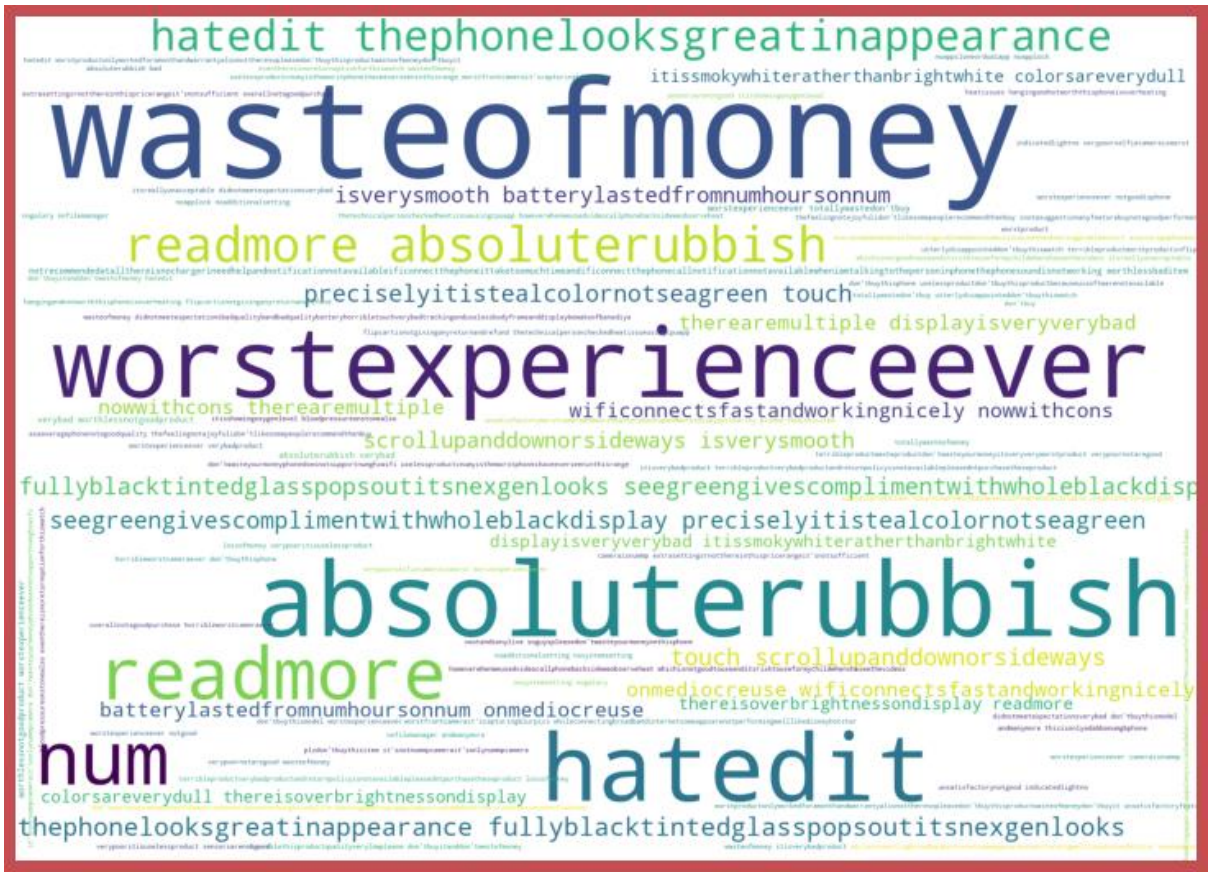
Top 30 most frequently used words:



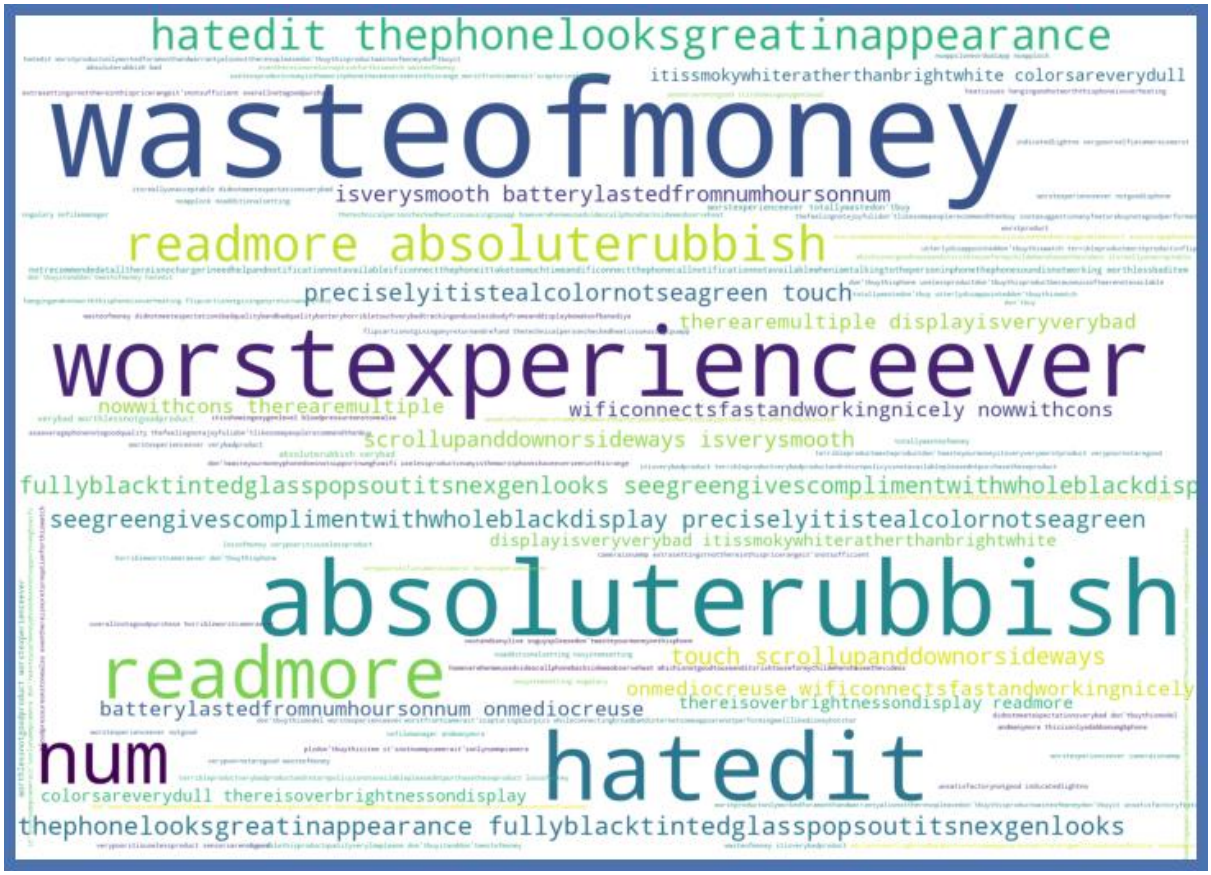
Top 30 rarely used words:



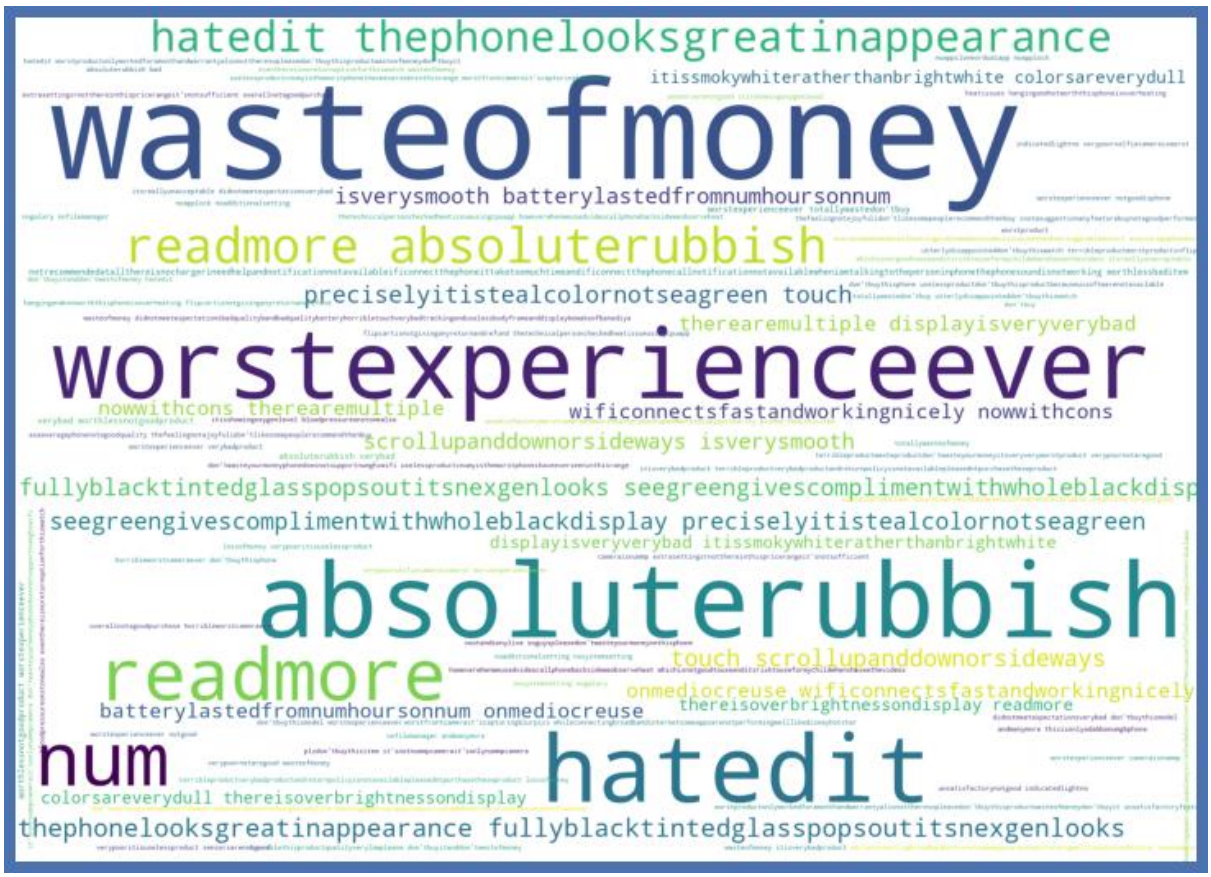
WordCloud for rating1:



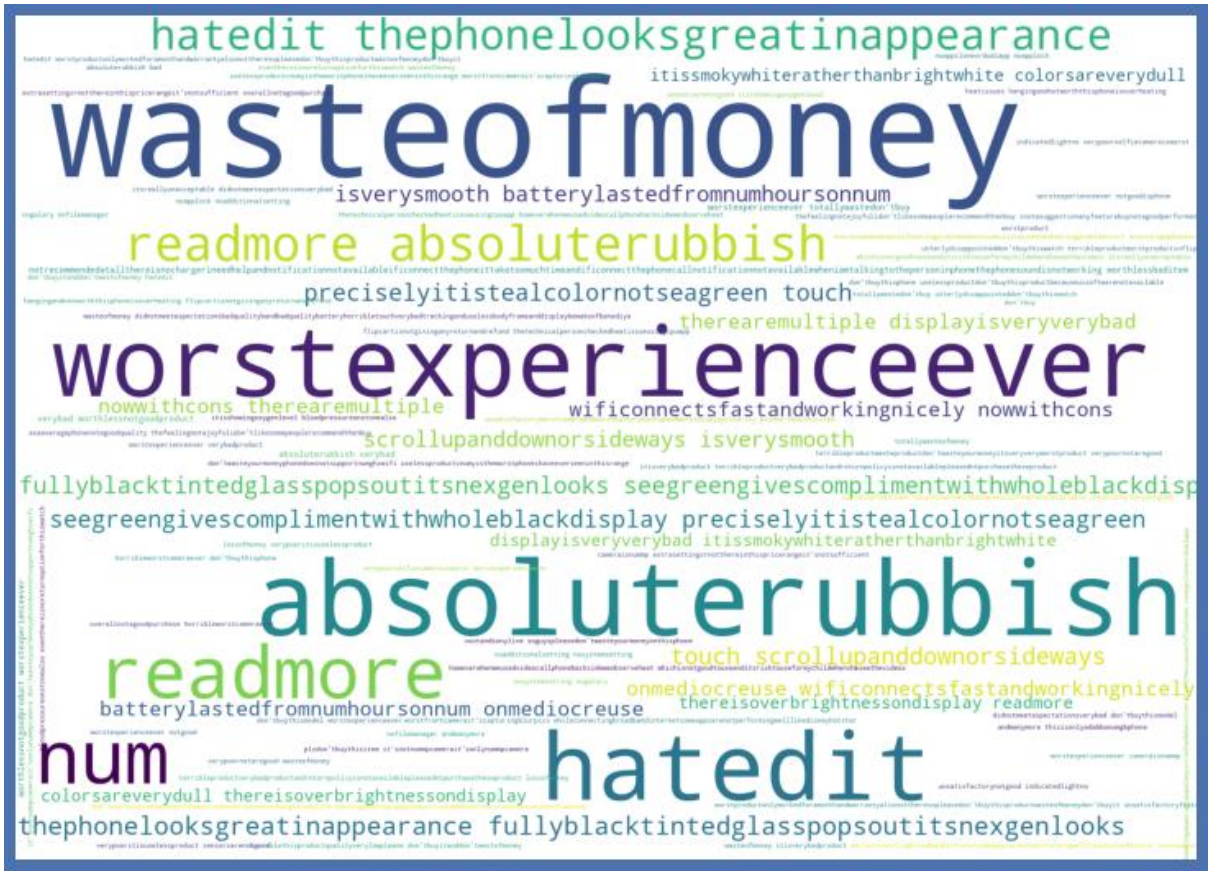
WordCloud for rating2:



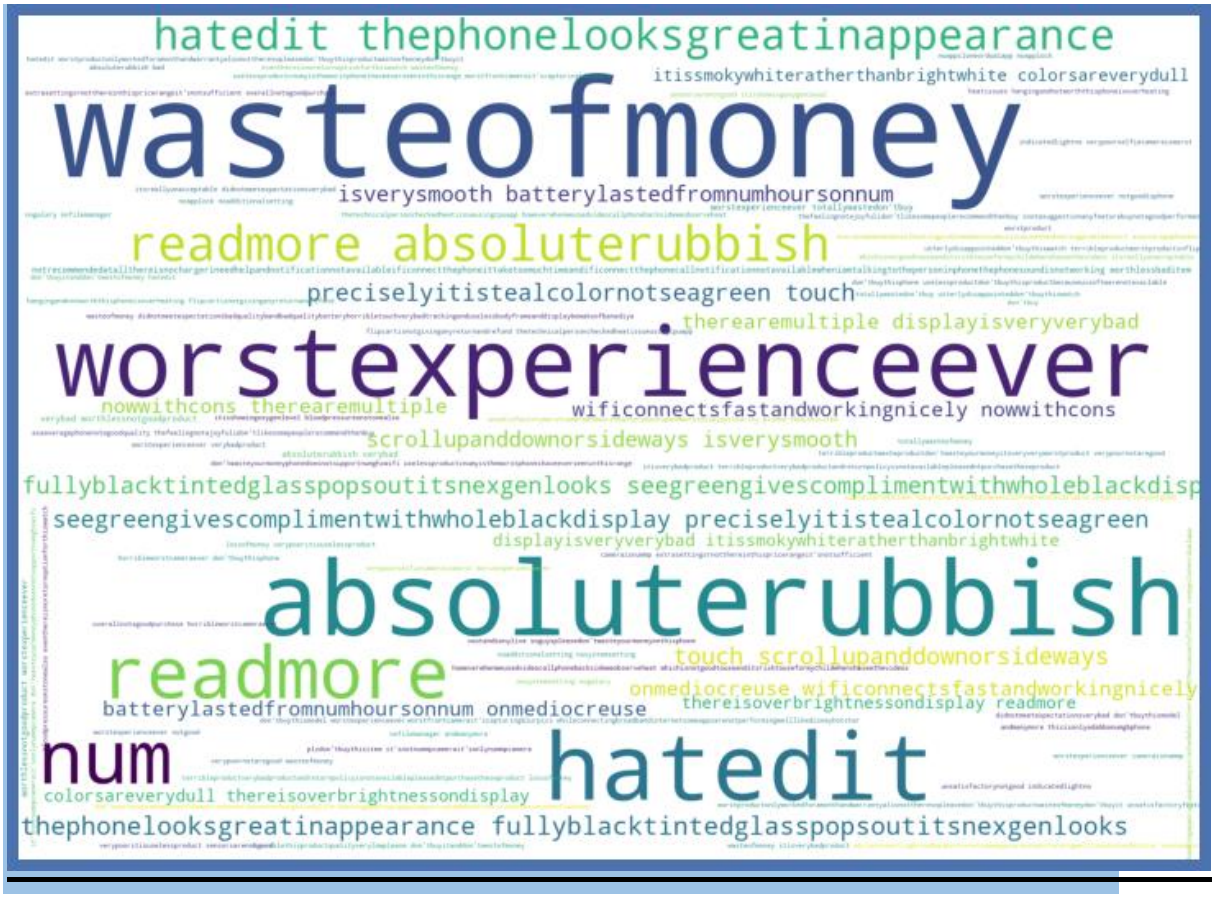
WordCloud for rating3:



WordCloud for rating4:



WordCloud for rating5:



Model Building:

Model:

Classification model is used. Classification refers to a predictive modeling problem where a class label is predicted for a given set of input data. It is a supervised technique. Regression models a target prediction based on independent variables.

Logistic Regression:

Linear regression is a

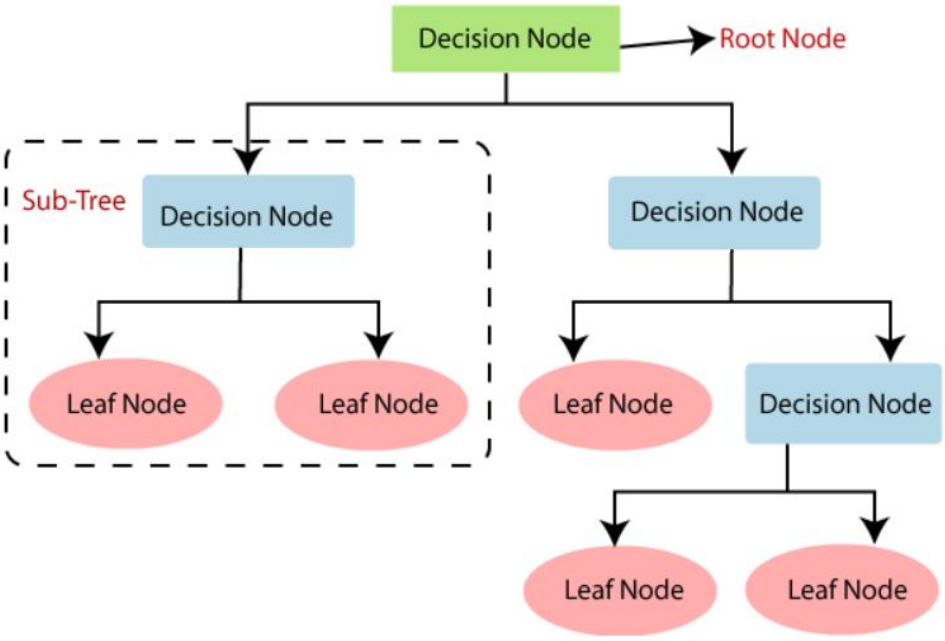
learning algorithm based on supervised learning. Logistic regression is a classification algorithm, used when the value of the target variable is categorical in nature. It is most commonly used when the data has binary input, so when it belongs to one class or the other, or is either a 0 or 1.

Decision Tree Classifier:

It is a tree-structured classifier, where internal nodes represent the features of a dataset, branches represent the decision rules and each leaf node represents the outcome.

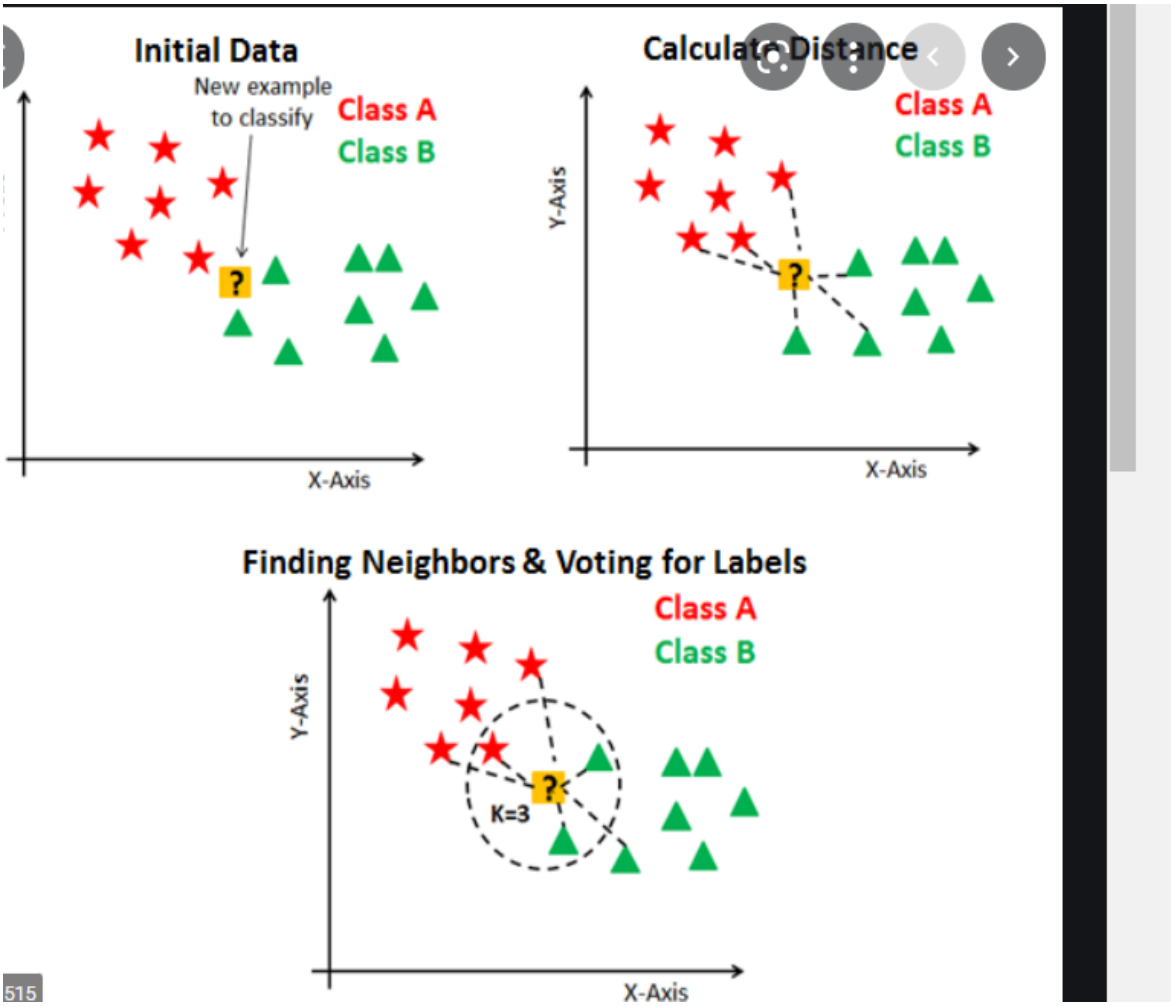
In a decision tree, there are two nodes, which are the decision node and leaf node. Decision nodes are used to make any decision and have multiple branches, whereas leaf nodes are the output of those decisions and do not contain any further branches.

It is called a decision tree because, similar to a tree, it starts with the root node, which expands on further branches and constructs a tree-like-structure.



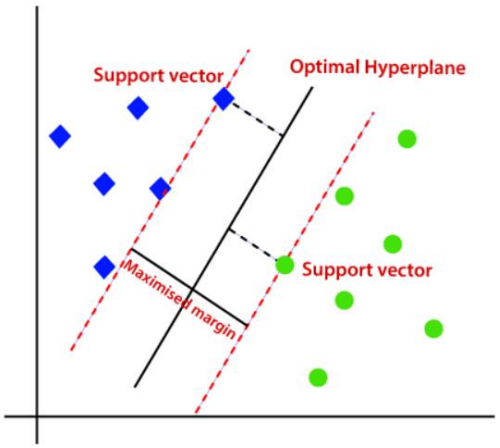
KNNeighbors Classifier:

The K in the name of this classifier represents the k nearest neighbors, where k is an integer value specified by the user. Hence as the name suggests, this classifier implements learning based on the k nearest neighbors. The choice of the value of k is dependent on data. By default, the KNeighborsClassifier looks for the 5 nearest neighbors. We must explicitly tell the classifier to use Euclidean distance for determining the proximity between neighboring points. Using our newly trained model, we predict whether a tumor is benign or not given its mean compactness and area.



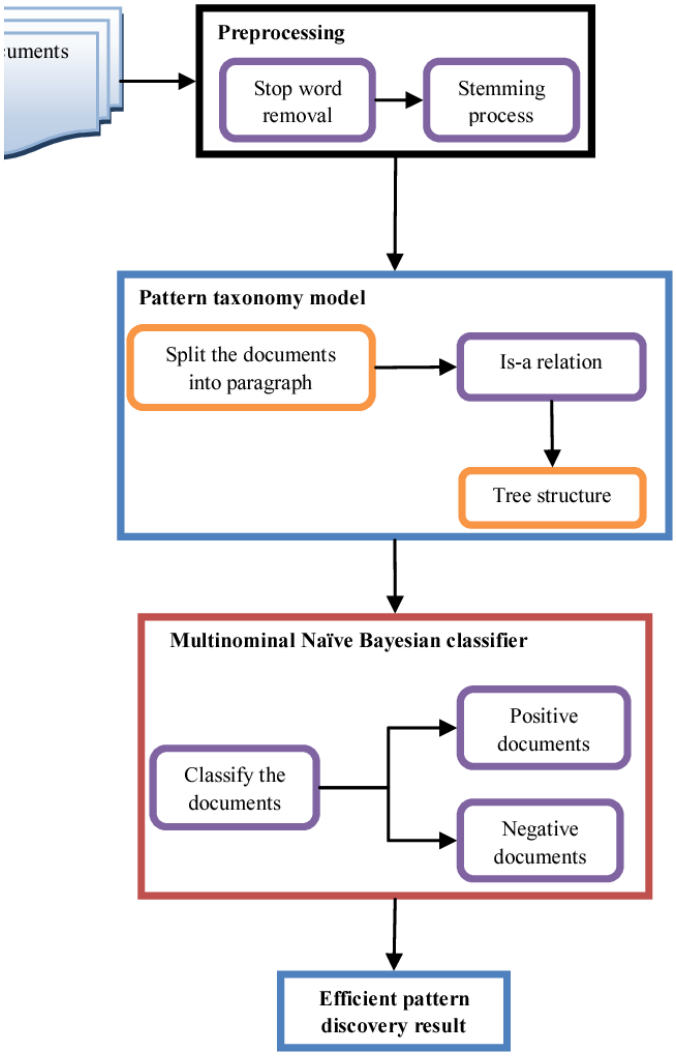
Support Vector Classifier(SVC):

A support vector machine (SVM) is a supervised machine learning model that uses classification algorithms for two-group classification problems. After giving an SVM model sets of labeled training data for each category, they're able to categorize new text.



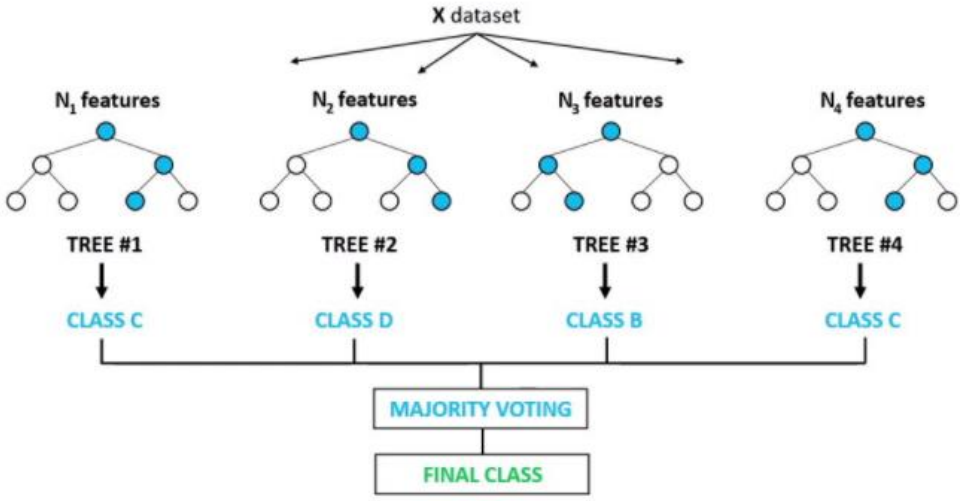
MultinomialNB:

Multinomial Naive Bayes algorithm is a probabilistic learning method that is mostly used in Natural Language Processing (NLP). ... Naive Bayes classifier is a collection of many algorithms where all the algorithms share one common principle, and that is each feature being classified is not related to any other feature.



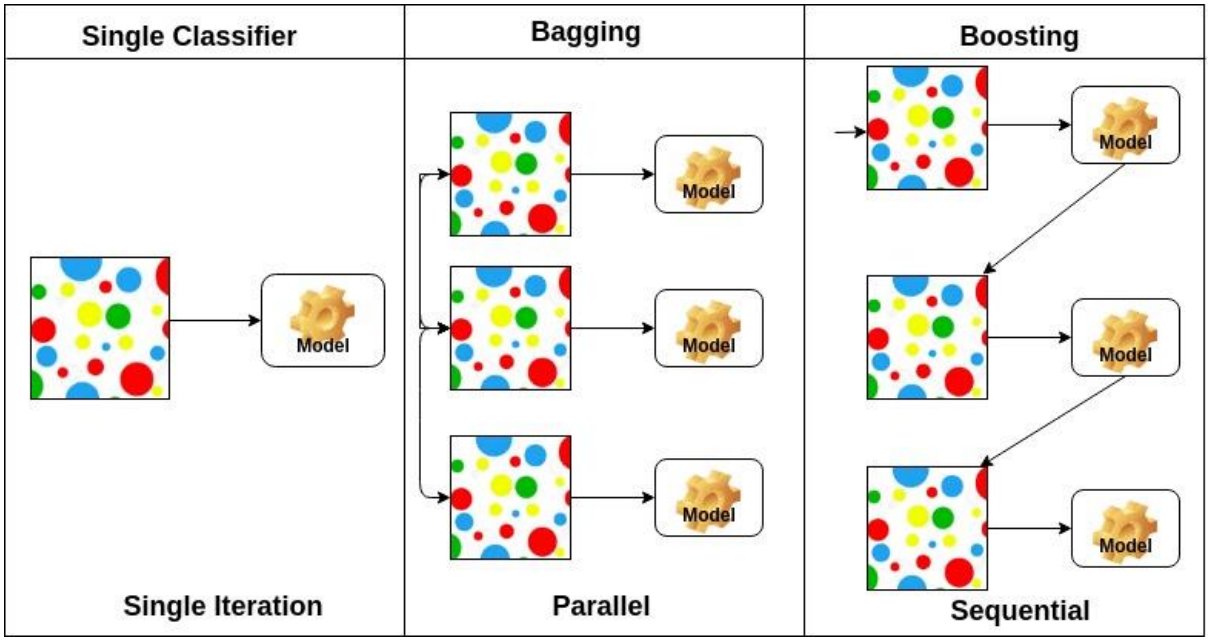
Random Forest Classifier:

It is a classification algorithm consisting of many decision trees. It uses bagging and feature randomness when building each individual tree and try to create an uncorrelated forest of trees whose prediction by committee is more accurate than that of any individual tree.



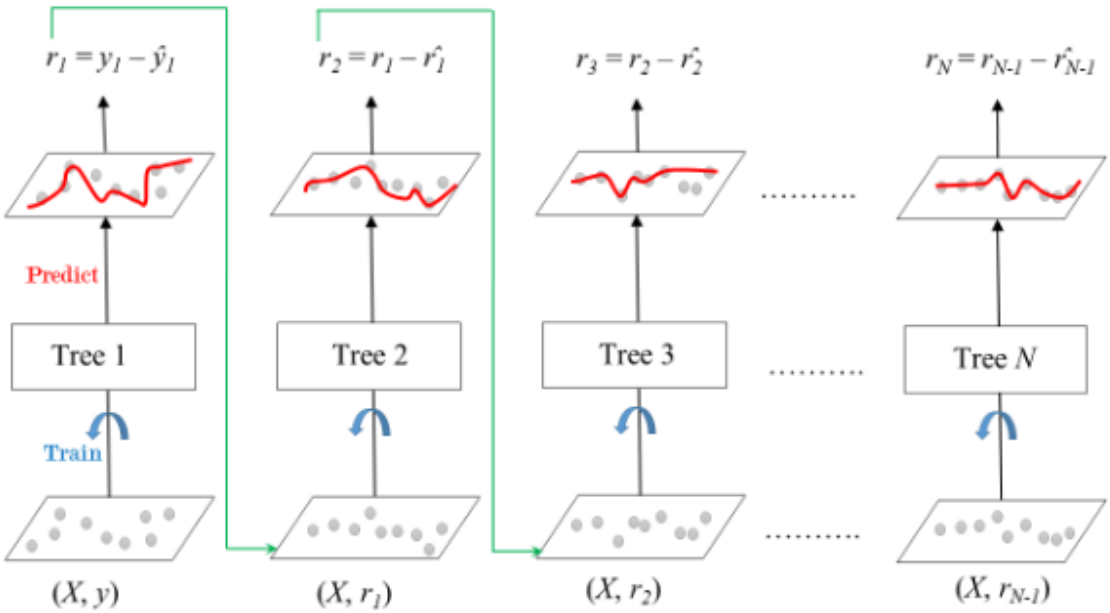
Ada Boost Classifier:

An AdaBoost classifier is a meta-estimator that begins by fitting a classifier on the original dataset and then fits additional copies of the classifier on the same dataset but where the weights of incorrectly classified instances are adjusted such that subsequent classifiers focus more on difficult cases.



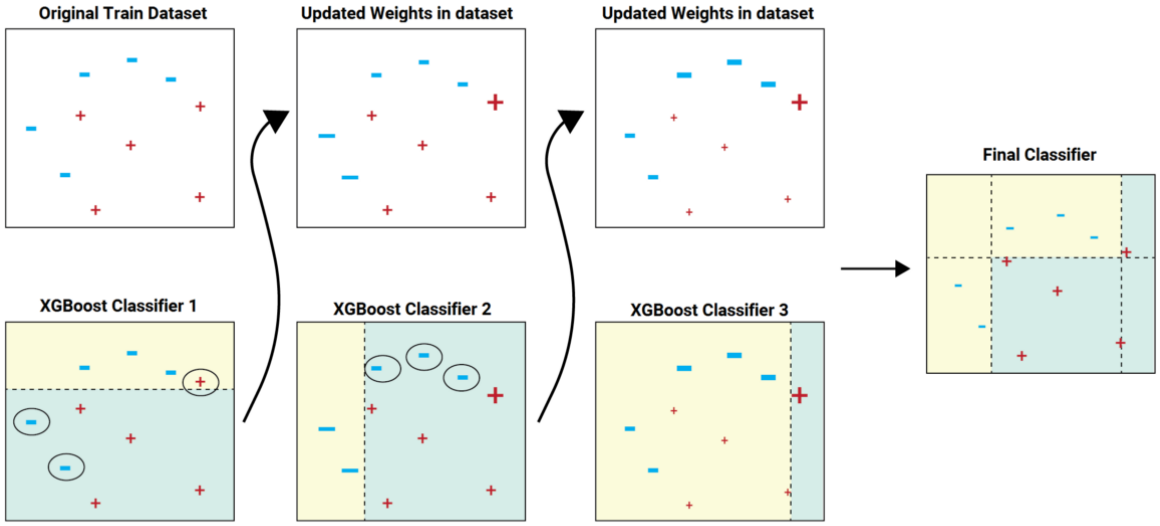
Gradient Boosting Classifier:

Gradient boosting classifiers are a group of machine learning algorithms that combine many weak learning models together to create a strong predictive model. Decision trees are usually used when doing gradient boosting.



XGBoost Classifier:

XGBoost provides a wrapper class to allow models to be treated like classifiers or regressors in the scikit-learn framework. The XGBoost model for classification is called XGBClassifier. We can create and and fit it to our training dataset. Models are fit using the scikit-learn API and the model. fit() function.



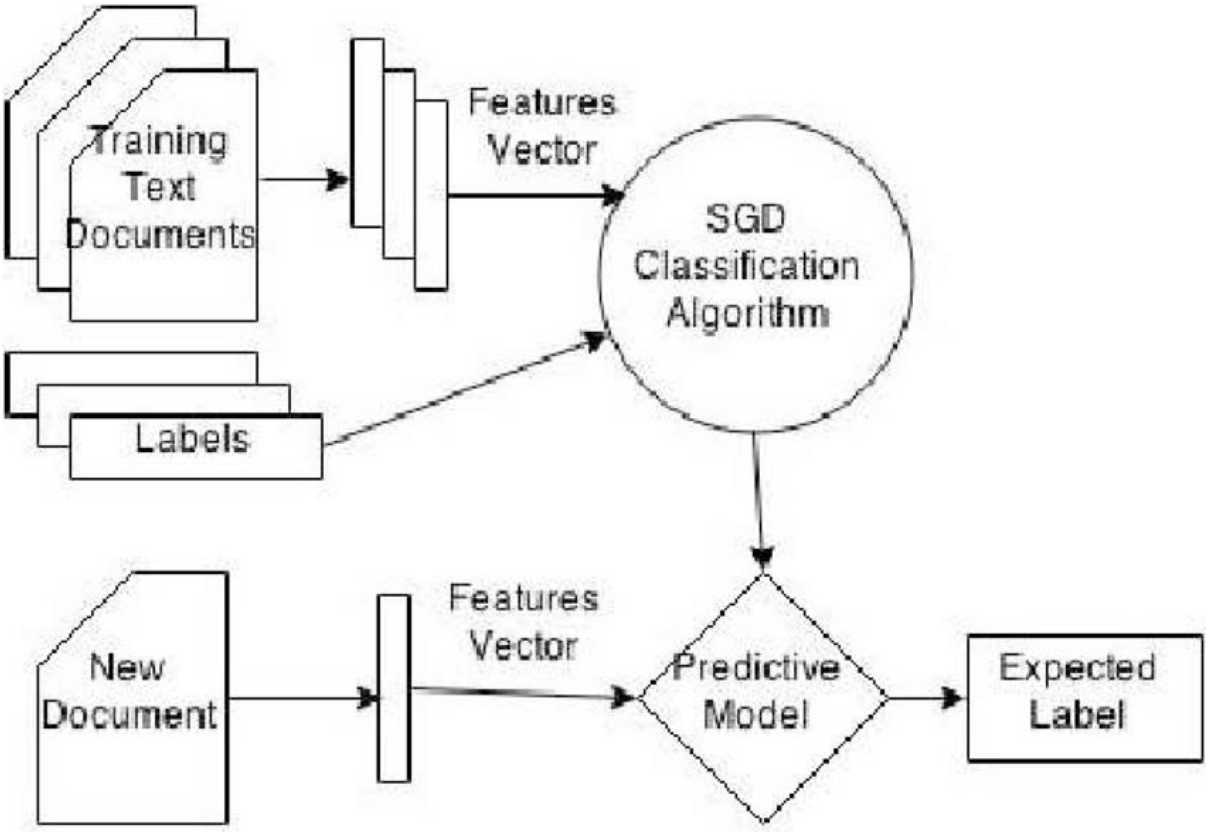
SGD Classifier:

This estimator implements regularized linear models with stochastic gradient descent (SGD) learning: the gradient of the loss is estimated each sample at a time and the model is updated along the way with a decreasing strength

schedule (aka learning rate). SGD allows minibatch (online/out-of-core) learning via the partial_fit method. For best results using the default learning rate schedule, the data should have zero mean and unit variance.

This implementation works with data represented as dense or sparse arrays of floating point values for the features. The model it fits can be controlled with the loss parameter; by default, it fits a linear support vector machine (SVM).

The regularizer is a penalty added to the loss function that shrinks model parameters towards the zero vector using either the squared euclidean norm L2 or the absolute norm L1 or a combination of both (Elastic Net). If the parameter update crosses the 0.0 value because of the regularizer, the update is truncated to 0.0 to allow for learning sparse models and achieve online feature selection.



Interpretation of Results:

Many machine learning algorithms are used to predict. Using these algorithms is beneficial so that the result can be as near to the claimed results. However, the prediction accuracy of these algorithms depends heavily on the given data

when training the model. If the data is in bad shape, the model will be overfitted and inefficient, which means that data pre-processing is an important part of this experiment and will affect the final results. Thus, multiple combinations of pre-processing methods need to be tested before getting the data ready to be used in training.

From Visualizations it is clear that how each feature from the dataset is relatable to the house price. In pre-processing, unnecessary feature is removed, outliers are removed and skewness from the input variables are removed. The data's are scaled and transformed for better training purpose. However, we have got 72% accuracy with all the models taken here, the chosen algorithm is Support Vector Classifier.

Saving the model:

I have saved my best model using .pkl

Model Saving:

```
import pickle
filename='rating prediction.pkl'
pickle.dump(svc,open(filename,'wb'))
```

4.CONCLUSION

In this project I have collected data of reviews and ratings for different products from flipkart.com.we have tried to detect the Ratings in commercial websites on a scale of 1 to 5 on the basis of the reviews given by the users. We made use of natural language processing and machine learning algorithms in order to do so.Then I have done different text processing for reviews column and chose equal number of text from each rating class to eliminate problem of imbalance. By doing different EDA steps I have analysed the text.We have checked frequently occurring words in our data as well as rarely occurring

words.After all these steps I have built function to train and test different algorithms and using various evaluation metrics I have selected SGDClassifier for our final model.Finally by doing hyperparameter tuning we got optimum parameters for our final model and the chosen best model is SupportVectorClassifier(SVC) with 72% accuracy.

5. Limitations

As we know the content of text in reviews is totally depends on the reviewer and they may rate differently which is totally depends on that particular person. So it is difficult to predict ratings based on the reviews with higher accuracies. Still we can improve our accuracy by fetching more data and by doing extensive hyper parameter tuning.

While we couldn't reach out goal of maximum accuracy in Ratings prediction project, we did end up creating a system that can with some improvement and deep learning algorithms get very close to that goal. As with any project there is room for improvement here. The very nature of this project allows for multiple algorithms to be integrated together as modules and their results can be combined to increase the accuracy of the final result. This model can further be improved with the addition of more algorithms into it. However, the output of these algorithms needs to be in the same format as the others. Once that condition is satisfied, the modules are easy to add as done in the code. This provides a great degree of modularity and versatility to the project.